**Module 9: Serverless Computing**

**Exercise 1: Implement Web Apps**

#### Task 1: Create an Azure web app

In this task, you will create an Azure web app.

1. Sign in to the [**Azure portal**](http://portal.azure.com/).
2. In the Azure portal, search for and select **App services**, and, on the **App Services** blade, click **+ Add**.
3. On the **Basics** tab of the **Web App** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Subscription | **the name of the Azure subscription you will be using in this lab** |
| Resource group | **the name of a new resource group az104-09a-rg1** |
| Web app name | **any globally unique name** |
| Publish | **Code** |
| Runtime stack | **PHP 7.4** |
| OS | **Windows** |
| Region | **the name of an Azure region where you can provision Azure apps** |
| App service plan | **S1** |

1. Click **Next: Monitoring >**, on the **Monitoring** tab of the **Web App** blade, set the **Enable Application Insights** switch to **No**, click **Review + create**, and then click **Create**.

**Note**: Typically, you would want to enable **Application Insights**, however, its functionality is not used in this lab.

**Note**: Wait until the web app is created before you proceed to the next task. This should take about a minute.

1. On the deployment blade, click **Go to resource**.

#### Task 2: Create a staging deployment slot

In this task, you will create a staging deployment slot.

1. On the blade of the newly deployed web app, click the **URL** link to display the default web page in a new browser tab.
2. Close the new browser tab and, back in the Azure portal, in the **Deployment** section of the web app blade, click **Deployment slots**.

**Note**: The web app, at this point, has a single deployment slot labeled **PRODUCTION**.

1. Click **+ Add slot**, and add a new slot with the following settings:

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Name | **staging** |
| Clone settings from | **Do not clone settings** |

1. Back on the **Deployment slots** blade of the web app, click the entry representing the newly created staging slot.

**Note**: This will open the blade displaying the properties of the staging slot.

1. Review the staging slot blade and note that its URL differs from the one assigned to the production slot.

#### Task 3: Configure web app deployment settings

In this task, you will configure web app deployment settings.

1. On the staging deployment slot blade, in the **Deployment** section, click **Deployment Center**.

**Note:** Make sure you are on the staging slot blade (rather than the production slot).

1. In the **Settings** on **Source** **- Continuous Deployment (CI/CD)** section, select **Local Git**, and then click **Save**.
2. Copy the resulting **Git Clone Url** to Notepad.

**Note:** You will need the Git Clone Url value in the next task of this lab.

1. Click **Local Git/FTPS credentials** toolbar icon to display **FTPS Credentials** pane, on **User scope**.
2. Complete the required information, and then click **Save**.

|  |  |
| --- | --- |
| **Setting** | **Value** |
| User name | **any unique name (must not contain @ character)** |
| Password | **any password that satisfies complexity requirements** |

**Note:** The password must be at least eight characters long, with two of the following three elements: letters, numbers, and non-alphanumeric characters.

**Note:** You will need these credentials in the next task of this lab.

#### Task 4: Deploy code to the staging deployment slot

In this task, you will deploy code to the staging deployment slot.

1. In the Azure portal, open the **Azure Cloud Shell** by clicking on the icon in the top right of the Azure Portal.
2. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.
3. From the Cloud Shell pane, run the following to clone the remote repository containing the code for the web app.

*git clone https://github.com/Azure-Samples/php-docs-hello-world*

1. From the Cloud Shell pane, run the following to set the current location to the newly created clone of the local repository containing the sample web app code.

*Set-Location -Path $HOME/php-docs-hello-world/*

1. From the Cloud Shell pane, run the following to add the remote git (make sure to replace the [deployment\_user\_name] and [git\_clone\_url] placeholders with the value of the **Deployment Credentials** user name and **Git Clone Url**, respectively, which you identified in previous task):

*git remote add [deployment\_user\_name] [git\_clone\_url]*

**Note**: The value following git remote add does not have to match the **Deployment Credentials** user name, but has to be unique

1. From the Cloud Shell pane, run the following to push the sample web app code from the local repository to the Azure web app staging deployment slot (make sure to replace the [deployment\_user\_name] placeholder with the value of the **Deployment Credentials** user name, which you identified in previous task):

*git push [deployment\_user\_name] master*

1. If prompted to authenticate, type the [deployment\_user\_name] and the corresponding password (which you set in the previous task).
2. Close the Cloud Shell pane.
3. On the staging slot blade, click **Overview** and then click the **URL** link to display the default web page in a new browser tab.
4. Verify that the browser page displays the **Hello World!** message and close the new tab.

#### Task 5: Swap the staging slots

In this task, you will swap the staging slot with the production slot

1. Navigate back to the blade displaying the production slot of the web app.
2. In the **Deployment** section, click **Deployment slots** and then, click **Swap** toolbar icon.
3. On the **Swap** blade, review the default settings and click **Swap**.
4. Click **Overview** on the production slot blade of the web app and then click the **URL** link to display the web site home page in a new browser tab.
5. Verify the default web page has been replaced with the **Hello World!** page.

#### Task 6: Configure and test autoscaling of the Azure web app

In this task, you will configure and test autoscaling of Azure web app.

1. On the blade displaying the production slot of the web app, in the **Settings** section, click **Scale out (App Service plan)**.
2. Click **Custom autoscale**.
3. Leave the default option **Scale based on a metric** selected and click **+ Add a rule**
4. On the **Scale rule** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Metric source | **Current resource** |
| Time aggregation | **Maximum** |
| Metric namespace | **App Service plans standard metrics** |
| Metric name | **CPU Percentage** |
| Operator | **Greater than** |
| Metric threshold to trigger scale action | **10** |
| Duration (in minutes) | **1** |
| Time grain statistic | **Maximum** |
| Operation | **Increase count by** |
| Instance count | **1** |
| Cool down (minutes) | **5** |

**Note**: Obviously these values do not represent a realistic configuration, since their purpose is to trigger autoscaling as soon as possible, without extended wait period.

1. Click **Add** and, back on the App Service plan scaling blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Instance limits Minimum | **1** |
| Instance limits Maximum | **2** |
| Instance limits Default | **1** |

1. Click **Save**.
2. In the Azure portal, open the **Azure Cloud Shell** by clicking on the icon in the top right of the Azure Portal.
3. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.
4. From the Cloud Shell pane, run the following to identify the URL of the Azure web app.

*$rgName = 'az104-09a-rg1'*

*$webapp = Get-AzWebApp -ResourceGroupName $rgName*

1. From the Cloud Shell pane, run the following to start and infinite loop that sends the HTTP requests to the web app:

*while ($true) { Invoke-WebRequest -Uri $webapp.DefaultHostName }*

1. Minimize the Cloud Shell pane (but do not close it) and, on the web app blade, in the **Monitoring** section, click **Process explorer**.

**Note**: Process explorer facilitates monitoring the number of instances and their resource utilization.

1. Monitor the utilization and the number of instances for a few minutes.

**Note**: You may need to **Refresh** the page.

1. Once you notice that the number of instances has increased to 2, reopen the Cloud Shell pane and terminate the script by pressing **Ctrl+C**.
2. Close the Cloud Shell pane.

#### Clean up resources

**Note**: Remember to remove any newly created Azure resources that you no longer use. Removing unused resources ensures you will not see unexpected charges.

1. In the Azure portal, open the **PowerShell** session within the **Cloud Shell** pane.
2. List all resource groups created throughout the labs of this module by running the following command:

*Get-AzResourceGroup -Name 'az104-09a\*'*

1. Delete all resource groups you created throughout the labs of this module by running the following command:

*Get-AzResourceGroup -Name 'az104-09a\*' | Remove-AzResourceGroup -Force -AsJob*

**Exercise 2: Implement Azure Container Instances**

#### Task 1: Deploy a Docker image by using the Azure Container Instance

In this task, you will create a new container instance for the web application.

1. Sign in to the [Azure portal](https://portal.azure.com/).
2. In the Azure portal, search for locate **Container instances** and then, on the **Container instances** blade, click **+ Add**.
3. On the **Basics** tab of the **Create container instance** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Subscription | **the name of the Azure subscription you are using in this lab** |
| Resource group | the name of a new resource group **az104-09b-rg1** |
| Container name | **az104-9b-c1** |
| Region | **the name of a region where you can provision Azure container instances** |
| Image Source | **Quickstart images** |
| Image | **microsoft/aci-helloworld (Linux)** |

1. Click **Next: Networking >** and, on the **Networking** tab of the **Create container instance** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| DNS name label | **any valid, globally unique DNS host name** |

**Note**: Your container will be publicly reachable at dns-name-label.region.azurecontainer.io. If you receive a **DNS name label not available** error message, specify a different value.

1. Click **Next: Advanced >**, review the settings on the **Advanced** tab of the **Create container instance** blade without making any changes, click **Review + Create**, and then click **Create**.

**Note**: Wait for the deployment to complete. This should take about 3 minutes.

**Note**: While you wait, you may be interested in viewing the [code behind the sample application](https://github.com/Azure-Samples/aci-helloworld). To view it, browse the \app folder.

#### Task 2: Review the functionality of the Azure Container Instance

In this task, you will review the deployment of the container instance.

1. On the deployment blade, click the **Go to resource** link.
2. On the **Overview** blade of the container instance, verify that **Status** is reported as **Running**.
3. Copy the value of the container instance **FQDN**, open a new browser tab, and navigate to the corresponding URL.
4. Verify that the **Welcome to Azure Container Instance** page is displayed.
5. Close the new browser tab, back in the Azure portal, in the **Settings** section of the container instance blade, click **Containers**, and then click **Logs**.
6. Verify that you see the log entries representing the HTTP GET request generated by displaying the application in the browser.

#### Clean up resources

1. In the Azure portal, open the **PowerShell** session within the **Cloud Shell** pane.
2. List all resource groups created throughout the labs of this module by running the following command:

*Get-AzResourceGroup -Name 'az104-09b\*'*

1. Delete all resource groups you created throughout the labs of this module by running the following command:

*Get-AzResourceGroup -Name 'az104-09b\*' | Remove-AzResourceGroup -Force -AsJob*

**Exercise 3: Implement Azure Kubernetes Service**

#### Task 1: Register the Microsoft.Kubernetes and Microsoft.KubernetesConfiguration resource providers.

In this task, you will register resource providers necessary to deploy an Azure Kubernetes Services cluster.

1. Sign in to the [Azure portal](https://portal.azure.com/).
2. In the Azure portal, open the **Azure Cloud Shell** by clicking on the icon in the top right of the Azure Portal.
3. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.
4. From the Cloud Shell pane, run the following to register the Microsoft.Insights and Microsoft.AlertsManagement resource providers.

*Register-AzResourceProvider -ProviderNamespace Microsoft.Kubernetes*

*Register-AzResourceProvider -ProviderNamespace Microsoft.KubernetesConfiguration*

1. Close the Cloud Shell pane.

#### Task 2: Deploy an Azure Kubernetes Service cluster

In this task, you will deploy an Azure Kubernetes Services cluster by using the Azure portal.

1. In the Azure portal, search for locate **Kubernetes services** and then, on the **Kubernetes services** blade, click **+ Add**.
2. On the **Basics** tab of the **Create Kubernetes cluster** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Subscription | **the name of the Azure subscription you are using in this lab** |
| Resource group | the name of a new resource group **az104-09c-rg1** |
| Kubernetes cluster name | **az104-9c-aks1** |
| Region | **the name of a region where you can provision Azure container instances** |
| Kubernetes version | **accept the default** |
| Node size | **accept the default** |
| Node count | **1** |

1. Click **Next: Node Pools >** and, on the **Node Pools** tab of the **Create Kubernetes cluster** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Virtual nodes | **Disabled** |

1. Click **Next: Authentication >** and, on the **Authentication** tab of the **Create Kubernetes cluster** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Service principal | **accept the default** |
| Enable RBAC | **Yes** |

1. Click **Next: Networking >** and, on the **Networking** tab of the **Create Kubernetes cluster** blade, specify the following settings (leave others with their default values):

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Network configuration | **Advanced** |
| DNS name prefix | any valid, globally unique DNS host name |
| Load balancer | **Standard** |
| HTTP application routing | **No** |

1. Click **Next: Integration >**, on the **Integration** tab of the **Create Kubernetes cluster** blade, set **Container monitoring** to **Disabled**, click **Review + create** and then click **Create**.

**Note**: In production scenarios, you would want to enable monitoring. Monitoring is disabled in this case since it is not covered in the lab.

**Note**: Wait for the deployment to complete. This should take about 10 minutes.

#### Task 3: Deploy pods into the Azure Kubernetes Service cluster

In this task, you will deploy a pod into the Azure Kubernetes Service cluster.

1. On the deployment blade, click the **Go to resource** link.
2. On the **az104-9c-aks1** Kubernetes service blade, in the **Settings** section, click **Node pools**.
3. On the **az104-9c-aks1 - Node pools** blade, verify that the cluster consists of a single pool with one node.
4. In the Azure portal, open the **Azure Cloud Shell** by clicking on the icon in the top right of the Azure Portal.
5. Switch the **Azure Cloud Shell** to **Bash** (black background).
6. From the Cloud Shell pane, run the following to retrieve the credentials to access the AKS cluster:

*RESOURCE\_GROUP='az104-09c-rg1'*

*AKS\_CLUSTER='az104-9c-aks1'*

*az aks get-credentials --resource-group $RESOURCE\_GROUP --name $AKS\_CLUSTER*

1. From the **Cloud Shell** pane, run the following to verify connectivity to the AKS cluster:

*kubectl get nodes*

1. In the **Cloud Shell** pane, review the output and verify that the one node which the cluster consists of at this point is reporting the **Ready** status.
2. From the **Cloud Shell** pane, run the following to deploy the **nginx** image from the Docker Hub:

*kubectl create deployment nginx-deployment --image=nginx*

**Note**: Make sure to use lower case letters when typing the name of the deployment (nginx-deployment)

1. From the **Cloud Shell** pane, run the following to verify that a Kubernetes pod has been created:

*kubectl get pods*

1. From the **Cloud Shell** pane, run the following to identify the state of the deployment:

*kubectl get deployment*

1. From the **Cloud Shell** pane, run the following to make the pod available from Internet:

*kubectl expose deployment nginx-deployment --port=80 --type=LoadBalancer*

1. From the **Cloud Shell** pane, run the following to identify whether a public IP address has been provisioned:

*kubectl get service*

1. Re-run the command until the value in the **EXTERNAL-IP** column for the **nginx-deployment** entry changes from **<pending>** to a public IP address. Note the public IP address in the **EXTERNAL-IP** column for **nginx-deployment**.
2. Open a browser window and navigate to the IP address you obtained in the previous step. Verify that the browser page displays the **Welcome to nginx!** message.

#### Task 4: Scale containerized workloads in the Azure Kubernetes service cluster

In this task, you will scale horizontally the number of pods and then number of cluster nodes.

1. From the **Cloud Shell** pane, run the following to scale the deployment by increasing of the number of pods to 2:

*kubectl scale --replicas=2 deployment/nginx-deployment*

1. From the **Cloud Shell** pane, run the following to verify the outcome of scaling the deployment:

*kubectl get pods*

**Note**: Review the output of the command and verify that the number of pods increased to 2.

1. From the **Cloud Shell** pane, run the following to scale out the cluster by increasing the number of nodes to 2:

*az aks scale --resource-group $RESOURCE\_GROUP --name $AKS\_CLUSTER --node-count 2*

**Note**: Wait for the provisioning of the additional node to complete. This might take about 3 minutes. If it fails, rerun the az aks scale command.

1. From the **Cloud Shell** pane, run the following to verify the outcome of scaling the cluster:

*kubectl get nodes*

**Note**: Review the output of the command and verify that the number of nodes increased to 2.

1. From the **Cloud Shell** pane, run the following to scale the deployment:

*kubectl scale --replicas=10 deployment/nginx-deployment*

1. From the **Cloud Shell** pane, run the following to verify the outcome of scaling the deployment:

*kubectl get pods*

**Note**: Review the output of the command and verify that the number of pods increased to 10.

1. From the **Cloud Shell** pane, run the following to review the pods distribution across cluster nodes:

*kubectl get pod -o=custom-columns=NODE:.spec.nodeName,POD:.metadata.name*

**Note**: Review the output of the command and verify that the pods are distributed across both nodes.

1. From the **Cloud Shell** pane, run the following to delete the deployment:

*kubectl delete deployment nginx-deployment*

1. Close the **Cloud Shell** pane.

#### Clean up resources

1. In the Azure portal, open the **Bash** shell session within the **Cloud Shell** pane.
2. List all resource groups created throughout the labs of this module by running the following command:

*az group list --query "[?starts\_with(name,'az104-09c')].name" --output tsv*

1. Delete all resource groups you created throughout the labs of this module by running the following command:

*az group list --query "[?starts\_with(name,'az104-09c')].[name]" --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'*